ISSN. 1412-0100 VOL 16, NO 1, APRIL 2015

Pengembangan Aplikasi *Steganografi* pada Citra dengan Metode *Blowfish* dan *Sequential Colour Cycle*

Ng Poi Wong[1], Sunario Megawan[2], Ade Wibowo Giri[3], Ayu Yolanda Nasution[4]
STMIK Mikroskil, Jl. Thamrin No. 112, 124, 140, Telp. (061) 4573767, Fax. (061) 4567789
[1,2,3,4]Jurusan Teknik Informatika, STMIK Mikroskil, Medan
[1]poiwong@mikroskil.ac.id, [2]sunario@mikroskil.ac.id, [3]101111309@students.mikroskil.ac.id,
[4]101110681@mikroskil.ac.id

Abstrak

Perkembangan teknologi yang semakin pesat, membuat sebuah perjalanan data melalui internet menjadi tidak aman. Untuk menjaga keamanan sebuah data tersebut, dibuatlah sebuah teknik pengamanan data menggunakan kriptografi. Namun, pengamanan dengan kriptografi masih membuat pihak ketiga menjadi curiga karena perubahan data masih terlihat. Untuk itu dilakukanlah kombinasi kriptografi dengan steganografi. Pada penelitian ini, algoritma kriptografi yang digunakan adalah algoritma Blowfish. Informasi yang telah dienkripsi dengan menggunakan algoritma Blowfish tersebut dimasukkan ke dalam media gambar dengan algoritma Steganografi, dalam hal ini algoritma steganografi yang digunakan adalah Sequential Colour Cycle. Kombinasi algoritma Blowfish dan Sequential Colour Cycle ini dapat memberikan peningkatan kualitas pengamanan pada suatu media gambar, sehingga informasi yang terdapat di dalam media gambar tidak diketahui oleh orang yang tidak di inginkan.

Kata kunci— keamanan data, kriptografi, steganografi, Blowfish, Sequential Colour Cycle

Abstract

Rapid technological developments make a transfer data via internet becomes unsafe. To maintain the data security, invented a technique using cryptography to secure the data. However, the cryptography security is still making the third party became suspicious because the changes of data are still detected, therefore, perform the combination of cryptography with steganography. In this research, the cryptography algorithm used is Blowfish algorithm. Information that has been encrypted using Blowfish algorithm will combine with image steganography algorithm, in this case the steganography algorithm used is Sequential Colour Cycle. The combination of the Blowfish algorithm and Sequential Colour Cycle can provide more security on a medium quality image, so that the information contained in the media image will not be known by unauthorized people.

Keywords— *data security, cryptography, steganography, blowfish, sequential colour cycle*

1. PENDAHULUAN

Pengamanan data menggunakan kriptografi saja tidak cukup, karena perubahan data masih terlihat dan sering membuat pihak ketiga menjadi curiga. Oleh karena itu dibuat sebuah aplikasi yang membuat algoritma steganografi teks pada media gambar menjadi lebih kuat dan aman dengan menambahkan algoritma kriptografi di dalamnya. Algoritma kriptografi yang digunakan adalah algoritma *Blowfish*. Dipilihnya algoritma *Blowfish* karena algoritma *Blowfish* masih lebih baik dibandingkan algortima lain seperti DES dan IDEA, sedangkan untuk mengetahui pesan asli yang dikeluarkan dari algoritma ini dibutuhkan banyak waktu dan usaha untuk melakukannya [1]. Informasi

yang terenkripsi menggunakan algoritma *Blowfish* tersebut kemudian dimasukkan ke dalam media gambar dengan menggunakan algoritma steganografi *Sequential Colour Cycle*. Dpilihnya algoritma *Sequential Colour Cycle* karena algoritma ini merupakan algoritma baru yang diajukan untuk mengoptimalkan mekanisme algoritma LSB (*Least Significant Bit*) dengan memanfaatkan warna RGB pada keseluruhan piksel dalam gambar, sehingga algoritma ini dapat menyandikan sampai 4 bit dalam setiap piksel RGB tanpa mempengaruhi kualitas dari media gambar yang digunakan [2].

Tujuan dari penelitian ini adalah untuk menghasilkan sebuah aplikasi penyembunyian pesan ke dalam media gambar dengan menggunakan algoritma *Blowfish* dan *Sequential Colour Cycle*, sedangkan manfaatnya agar diperolehnya peningkatan kualitas pengamanan sebuah data.

Adapun batasan masalah dari penelitian ini adalah informasi yang ingin disembunyikan berupa teks atau file TXT, panjang informasi yang ingin disembunyikan harus lebih kecil dari media yang digunakan untuk menyembunyikan informasi tersebut, media yang digunakan untuk menyembunyikan informasi dari aplikasi ini hanya media gambar. Format gambar penampung yang digunakan untuk menyembunyikan informasi dari aplikasi ini adalah file gambar berwarna BMP 24 bit, format gambar stego yang dihasilkan dari aplikasi ini adalah file gambar berwarna BMP 24 bit, jumlah panjang kunci yang digunakan paling sedikit 1 karakter dan paling banyak 56 karakter, perhitungan tingkat kualitas gambar yang belum disisipkan pesan dengan gambar yang sudah disisipkan pesan dengan cara mencari nilai PSNR dan MSE.

2. METODE PENELITIAN

2.1 Tinjauan Pustaka

2.1.1 *Blowfish*

Blowfish merupakan cipher blok, yang berarti selama proses enkripsi dan dekripsi, *Blowfish* akan membagi pesan menjadi blok-blok dengan ukuran yang sama panjang, yaitu 64 bit. Algoritma ini terdiri dari dua bagian yaitu perluasan kunci (*key expansion*) dan enkripsi data. *Key expansion* merupakan tahapan untuk memperluas kunci yang panjangnya 32 bit sampai 448 bit menjadi beberapa array subkunci (*subkey*) dengan total 4168 byte.

Enkripsi data terjadi pada jaringan *feistel* sebanyak 16 putaran. Setiap putaran terdiri dari permutasi kunci dan subtitusi data. Semua operasi yang digunakan adalah penambahan dan XOR pada variabel 32 bit. Tambahan operasi lainnya hanyalah empat penelusuran tabel (table lookup) untuk setiap putaran.

Blowfish menggunakan subkunci yang besar. Kunci ini harus dihitung sebelum enkripsi atau dekripsi data di mulai. Kunci tersebut terdiri dari :

1. Array P, terdiri dari delapan belas 32 bit subkunci: P1, P2, P3, . . . ,P18
2. Empat 32 bit S-box, masing-masing mempunyai 256 entri :
   S1[0], S1[1], . . . , S1[255], S2[0], S2[1], . . . , S2[255]
   S3[0], S3[1], . . . , S3[255], S4[0], S4[1], . . . , S4[255]

Nilai awal dari array P dan 4 *S-box* secara berurutan adalah string yang tetap, yang terdiri dari digit hexadesimal dari *phi* (π) [3].

2.1.2 *Sequential Colour Cycle*

Algoritma *Sequential Colour Cycle* dapat menyembunyikan 1 s/d 4 bit LSB pada setiap warna RGB dari piksel pada gambar, tergantung dari ukuran informasi yang akan di sembunyikan. Misalnya, jika ukuran informasi yang ingin di sembunyikan adalah 10% dari ukuran gambar penampung, maka akan digunakan 1 bit LSB pada setiap warna RGB dari piksel pada gambar. Jika ukuran informasi yang ingin di sembunyikan setengah dari ukuran gambar penampung, maka maksimal bit yang digunakan adalah 1 bit LSB pada setiap warna RGB dari piksel pada gambar.

Dengan menggunakan algoritma *Sequential Colour Cycle*, seluruh LSB sepenuhnya digunakan untuk menyembunyikan informasi dari setiap piksel pada gambar secara berurutan. Jadi, pesan dapat di sembunyikan dalam gambar penampung tanpa harus memperbesar ukuran dari gambar tersebut, karena LSB mengganti atau memanipulasi bit yang paling akhir, bukan menambahnya [2].

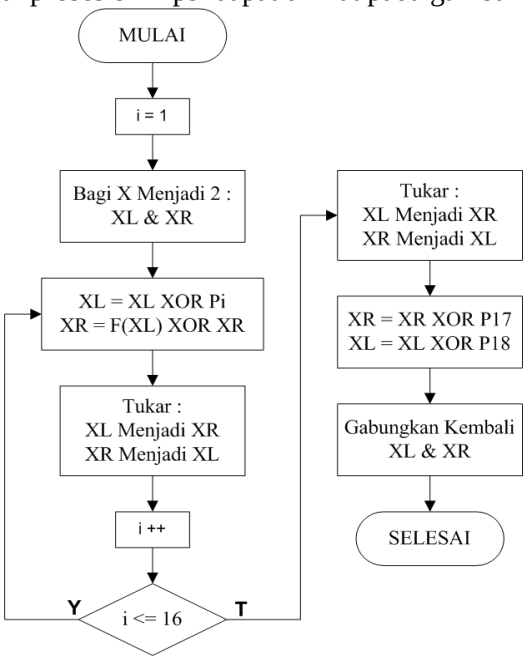
2.2 Analisis dan Perancangan

2.2.1 Analisis Algoritma *Blowfish*

Blowfish merupakan algoritma yang menerapkan jaringan *feistel* (*feistel network*) yang terdiri dari 16 putaran. Panjang dari pesan adalah 64 bit, di simbolkan menjadi X. Apabila panjang dari pesan kurang dari 64 bit, maka akan di tambahkan bit 0 sampai menjadi 64 bit. Berikut tahapan untuk melakukan proses enkripsi :

1. Bagi X menjadi 2, 32 bit pertama disebut XL, 32 bit yang kedua disebut XR.
2. Selanjutnya lakukan operasi XL = XL xor Pi dan XR = F(XL) xor XR.
3. Hasil dari operasi diatas ditukar, XL menjadi XR dan XR menjadi XL.
4. Lakukan sebanyak 16 kali, perulangan yang ke-16 lakukan lagi proses penukaran XL dan XR.
5. Pada proses ke-17 lakukan operasi untuk XR = XR xor P17 dan XL = XL xor P18.
6. Proses terakhir satukan kembali XL dan XR sehingga menjadi 64 bit kembali.

Untuk lebih jelasnya mengenai proses enkripsi dapat dilihat pada gambar 1 berikut.



Gambar 1. Flowchart proses enkripsi algoritma blowfish

Tahapan dalam fungsi F adalah :

1. Bagi XL menjadi empat bagian, masing-masing 8 bit : a, b, c, dan d.
2. Lakukan proses perhitungan sebagai berikut :

$$F(XL) = \left((S1[a] + S2[b] \bmod 2^{32}) xor\ S3[c]\right) + S4[d] \bmod 2^{32} \quad (1)$$

Tahapan dalam proses dekripsi sama persis dengan proses enkripsi, kecuali nilai dari P1, P2, P3, . . . , P18 digunakan pada urutan yang terbalik.

Metode untuk menghitung subkunci adalah sebagai berikut :

1. Pertama inisialisasi array P dan kemudian 4 S-box secara berurutan dengan string yang tetap. String ini terdiri dari digit hexadesimal dari phi (π). Kemudian inisialisasi sebuah string 64 bit yang berupa string 0.

2. XOR P1 dengan 32 bit pertama kunci, XOR P2 dengan 32 bit kedua dari kunci dan seterusnya untuk setiap bit dari kunci (sampai P18). Ulangi terhadap bit kunci sampai seluruh array P di XOR dengan bit kunci.
3. Enkrip semua string nol dengan algoritma *Blowfish* menggunakan subkunci seperti yang dijelaskan pada langkah 1 dan 2.
4. Ganti P1 dan P2 dengan keluaran dari langkah 3.
5. Enkrip keluaran dari langkah 3 dengan algoritma *Blowfish* dengan subkunci yang sudah dimodifikasi.
6. Ganti P3 dan P4 dengan keluaran dari langkah 5.
7. Lanjutkan proses tersebut, ganti seluruh elemen dari array P, kemudian seluruh ke-4 S-box berurutan, dengan keluaran yang berubah secara kontiniu dari algoritma *Blowfish*.

Total diperlukan 521 iterasi untuk menghasilkan semua subkunci yang dibutuhkan, kemudian aplikasi dapat menyimpan subkunci ini dan tidak dibutuhkan kembali langkah-langkah proses penurunan ini berulang kali, kecuali kunci yang digunakan berubah [3].

2.2.2 Analisis Algoritma *Sequential Colour Cycle*

Tahapan dalam proses encoding adalah :

1. Hitung ukuran dari informasi yang ingin di sembunyikan dan ukuran dari gambar penampung (cover image).
2. Tentukan total byte pada seluruh warna RGB dari piksel pada gambar penampung (cover image) yang dibutuhkan untuk menyembunyikan informasi tersebut.

$$bit\ per\ piksel = bit\ per\ byte \times 3 \quad (1)$$

$$jumlah\ piksel\ gambar\ penampung = \frac{(ukuran\ gambar\ penampung \times 8)}{bit\ per\ piksel} \quad (2)$$

$$jumlah\ piksel\ informasi = \frac{(ukuran\ informasi \times 8)}{bit\ per\ piksel} \quad (3)$$

3. Sistem membaca seluruh byte dari informasi rahasia yang ingin disembunyikan, kemudian konversi jumlah bit dari data rahasia pada persamaan (3) dari bilangan bulat ke dalam string biner.
4. Pisahkan nilai piksel menjadi tiga bagian menjadi warna RGB.
5. Dapatkan total berapa kali proses LSB yang dibutuhkan.
6. Hitung 7 array pertama piksel (dari piksel ke-0 sampai piksel ke-6) dari gambar penampung (*cover image*) untuk menyimpan detail dari pesan rahasia yang akan di jelaskan pada tabel 1. Jumlah piksel k yang digunakan untuk menyimpan nama file dari pesan rahasia ialah :

$$k = n - 7 \quad (4)$$

Keterangan :

k : Kapasitas dari nama file yang digunakan dalam satuan piksel.

n : Piksel k setelah piksel ke-6. Piksel yang tersisa dari gambar penampung (cover image) akan digunakan untuk menyimpan isi dari pesan rahasia.

Tabel 1. Detail piksel yang digunakan untuk menyimpan pesan rahasia

No	*Piksel (Array[])*	*Detail yang Disimpan*
1.	0	Jumlah LSB yang digunakan
2.	1	Panjang dari nama file
3.	2 sampai 6	Panjang dari pesan rahasia
4.	7 sampai n	Nama file dari pesan rahasia
5.	n + 1 sampai n + k, dimana k adalah kapasitas gambar penampung (*cover image*) dalam satuan piksel.	Isi dari pesan rahasia

7. Hitung jumlah LSB dari masing-masing warna piksel yang digunakan untuk menyembunyikan pesan rahasia. Sejak nilai RGB terbagi menjadi tiga bagian, algoritma ini hanya mengambil tiga nilai terakhir dari setiap byte, yaitu 216, 224, dan 232. Awalnya, bit yang tidak digunakan dalam

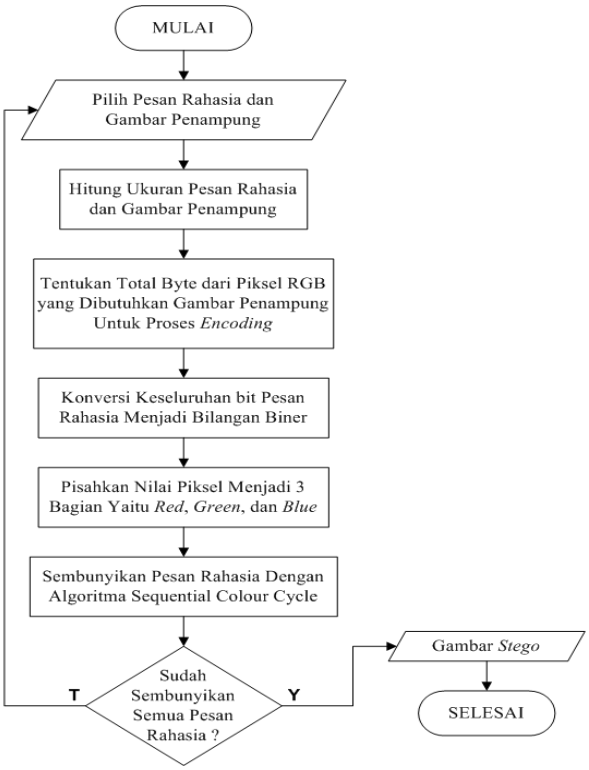
warna piksel RGB akan dibuang. Kemudian, bit yang tersisa akan digunakan untuk menyembunyikan isi pesan rahasia. Misalnya, jika dipilih gambar penampung adalah gambar yang mengandung warna RGB di setiap pikselnya, maka persamaan berikut dapat menghitung jumlah bit yang dibuang.

$$x = 8 - m \quad (5)$$

Keterangan :
x : Bit yang tidak digunakan setelah proses encoding.
m : Bit yang digunakan untuk mengkodekan data; m = 1, 2, 3, dan 4.
Setelah mendapatkan nilai bit yang tidak terpakai dari piksel RGB pada persamaan (5), nilai dari byte yang asli akan dikurangkan dengan nilai bit x yang tidak terpakai dimana x untuk mengidentifikasi bit yang dikodekan. Oleh karena itu, untuk menghitung nilai piksel yang baru dengan data yang dikodekan, nilai asli dari piksel dan bit yang dikodekan akan ditambahkan bersama-sama untuk mendapatkan piksel baru yang berisi data yang tersimpan. Perhitungan berikut adalah cara untuk memperoleh total LSB yang dikodekan dan nilai dari piksel baru dengan data yang dikodekan :

$$Total\ LSB\ yang\ dikodekan = (s_1 - x_1) + (s_2 - x_2) + (s_3 - x_3) \quad (6)$$

Nilai piksel baru dengan data yang dikodekan = byte asli + total LSB yang dikodekan.
Keterangan :
x1 : Bit yang tidak digunakan dari piksel merah.
x2 : Bit yang tidak digunakan dari piksel hijau.
x3 : Bit yang tidak digunakan dari piksel biru.
s1 : Bit yang asli dari piksel merah.
s2 : Bit yang asli dari piksel hijau.
s3 : Bit yang asli dari piksel biru.
Gambar 2 menunjukkan alur proses encoding dengan menggunakan algoritma *Sequential Colour Cycle*.

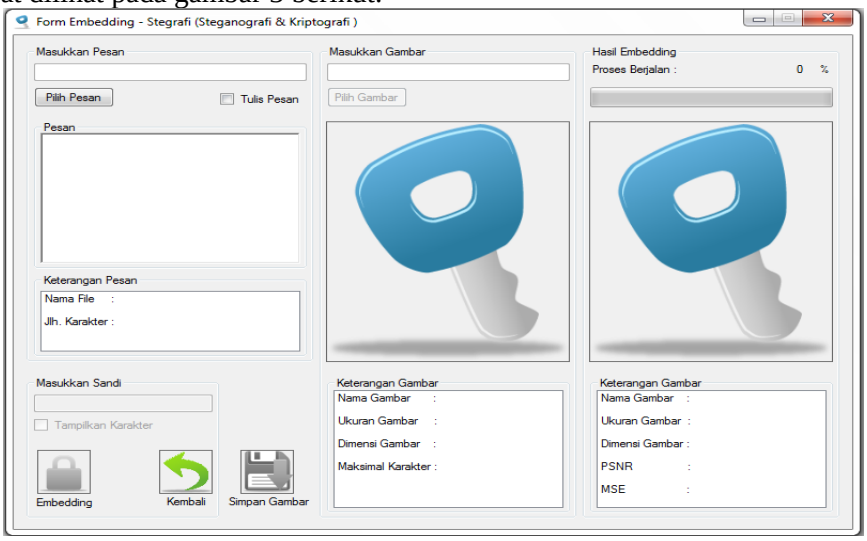


Gambar 2. Flowchart proses encoding algoritma *sequential colour cycle*

Tahapan untuk proses Decoding adalah kebalikan dari proses enkoding di atas.

3. HASIL DAN PEMBAHASAN

Tampilan form embedding merupakan tampilan form untuk melakukan enkripsi pesan dan kemudian menyisipkannya ke dalam gambar. Hasil dari tampilan form embedding ketika pertama dijalankan dapat dilihat pada gambar 3 berikut.

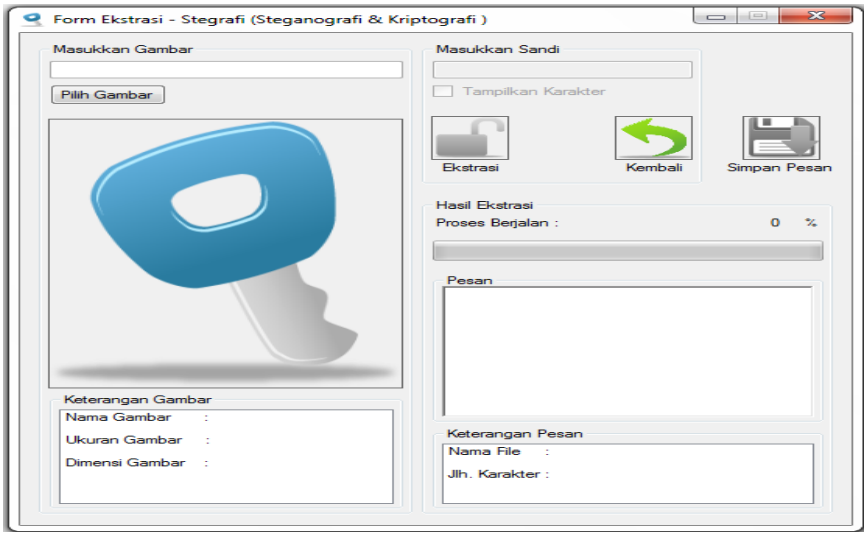


Gambar 3. Tampilan form embedding

Pada form embedding, proses awal yang harus dilakukan adalah memasukkan pesan yang ingin disembunyikan. Cara memasukkan pesan tersebut dapat dilakukan dengan dua cara, yaitu dengan menekan tombol pilih pesan atau dapat langsung ditulis dengan menekan check box tulis pesan. Setelah memasukkan pesan yang ingin disembunyikan, maka tombol pilih gambar akan aktif, dan pengguna dapat memilih media citra yang akan digunakan sebagai media penampung pesan yang disembunyikan.

Setelah memilih media citra, maka text box kata sandi dan check box tampilkan karakter akan aktif, dan pengguna dapat memasukkan sandi yang akan digunakan dalam proses embedding. Selanjutnya jika text box kata sandi telah diisi, maka icon embedding akan aktif dan proses penyembunyian pesan ke dalam gambar dapat dilakukan dengan menekan icon tersebut. Icon simpan gambar akan aktif setelah proses embedding selesai dilakukan.

Tampilan form ekstraksi merupakan tampilan form untuk mengekstrak pesan dari gambar kemudian melakukan proses dekripsi terhadap pesan. Hasil dari tampilan form ekstraksi ketika pertama dijalankan dapat dilihat pada gambar 4 berikut.



Gambar 4. Tampilan form ekstraksi

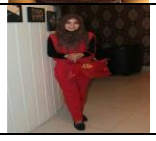
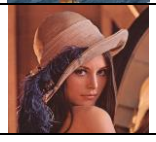
Pada form ekstraksi, proses awal yang harus dilakukan adalah memasukkan gambar yang sudah disisipkan pesan sebelumnya dengan cara menekan tombol pilih gambar. Setelah memilih gambar yang ingin diekstrak, maka textbox kata sandi dan check box tampilkan karakter akan aktif, dan pengguna dapat memasukkan sandi yang sama ketika melakukan proses embedding. Icon ekstrasi akan aktif setelah text box kata sandi diisi, dan pengguna dapat mengekstrak pesan yang terdapat dalam gambar tersebut dengan menekan icon dekrispi. Setelah proses ekstraksi selesai dilakukan, maka icon simpan pesan akan aktif.

Adapun pengujian yang dilakukan terhadap aplikasi yang dibangun, antara lain :

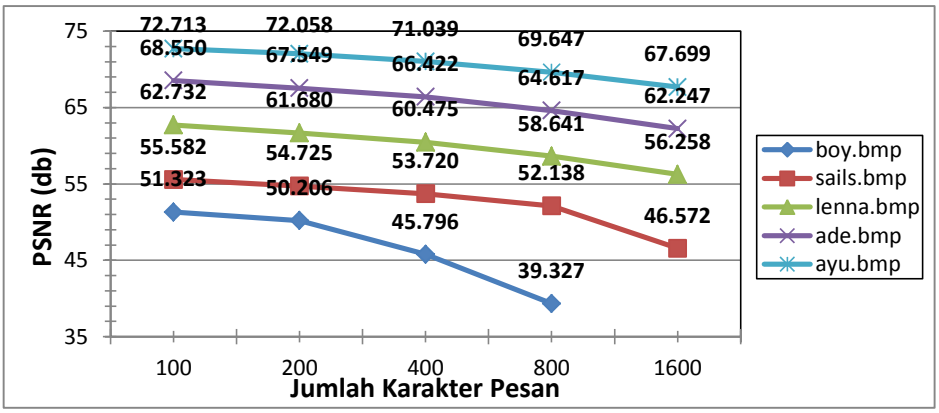
1. Mengukur tingkat kualitas dari gambar sebelum dan sesudah disisipkan pesan dengan kombinasi dari resolusi gambar yang bervariasi, jumlah karakter pesan yang bervariasi, dan jumlah panjang kunci yang bervariasi.
2. Mengembalikan pesan yang sudah disisipkan ke dalam gambar dengan menggunakan kunci yang sama.
3. Mengembalikan pesan yang sudah disisipkan ke dalam gambar dengan menggunakan kunci yang berbeda.

Tabel 2 menunjukkan kelima gambar penampung yang digunakan dalam proses pengujian.

Tabel 2. Kelima gambar penampung yang digunakan untuk pengujian

No.	*Nama Gambar*	*Gambar*	*Resolusi Gambar*	*No.*	*Nama Gambar*	*Gambar*	*Resolusi Gambar*
1.	boy.bmp		32 x 32 piksel	4.	ade.bmp		256 x 256 piksel
2.	sails.bmp		64 x 64 piksel	5.	ayu.bmp		512 x 512 piksel
3.	lenna.bmp		128 x 128 piksel				

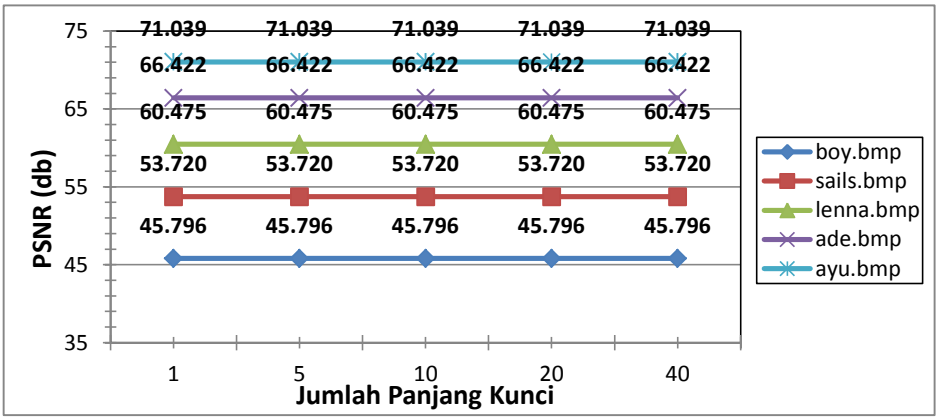
Pengujian pertama yang dilakukan adalah untuk mengukur tingkat kualitas dari gambar sebelum dan sesudah disisipkan pesan dengan resolusi gambar yang bervariasi dan jumlah karakter pesan yang bervariasi dengan panjang kunci 1 karakter. Hasil dari pengujian yang pertama dapat dilihat pada grafik gambar 5 berikut.



Gambar 5. Grafik nilai PSNR pengujian yang pertama

Dari hasil pengujian yang pertama, terdapat satu percobaan yang tidak dapat dilakukan karena gambar penampung tidak cukup menampung jumlah karakter pesan tersebut. Dari grafik dapat dilihat bahwa jumlah karakter pesan yang disisipkan pada setiap gambar penampung berpengaruh terhadap nilai PSNR yang dihasilkan atau dengan kata lain semakin banyak karakter pesan yang disisipkan ke dalam gambar panampung, maka semakin berkurang juga kualitas dari gambar yang dihasilkan. Hal ini ditandai dengan berkurangnya nilai PSNR yang dihasilkan oleh masing-masing gambar penampung yang disisipkan pesan dengan jumlah karakter pesan yang bervariasi dan jumlah panjang kunci 1 karakter.

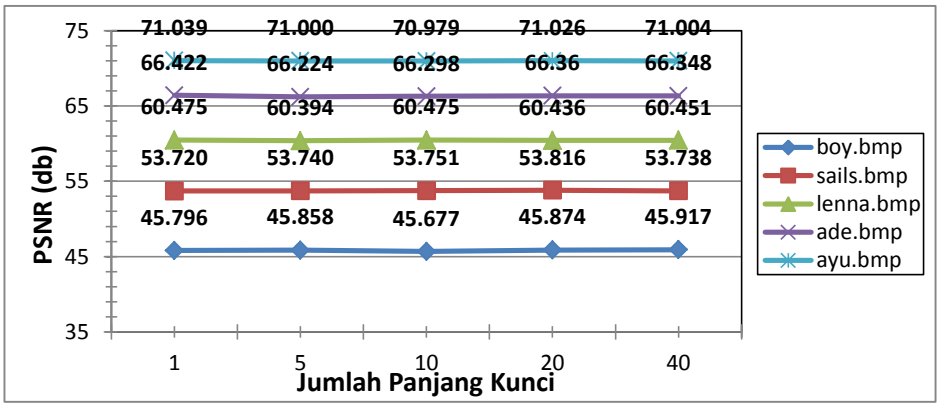
Pengujian kedua yang dilakukan adalah untuk mengukur tingkat kualitas dari gambar sebelum dan sesudah disisipkan pesan dengan resolusi gambar yang bervariasi dan jumlah karakter pesan 400 karakter dengan panjang kunci yang bervariasi dan merupakan karakter yang sama. Hasil dari pengujian yang kedua dapat dilihat pada grafik gambar 6 berikut.



Gambar 6. Grafik nilai PSNR pengujian yang kedua

Dari grafik tersebut dapat dilihat bahwa jumlah panjang kunci dengan karakter yang sama tidak berpengaruh terhadap nilai PSNR yang dihasilkan atau dengan kata lain tidak terjadi perubahan kualitas dari gambar yang dihasilkan. Hal ini dapat ditandai dengan tidak terjadinya perubahan nilai PSNR yang dihasilkan oleh masing-masing gambar.

Pengujian ketiga yang dilakukan adalah untuk mengukur tingkat kualitas dari gambar sebelum dan sesudah disisipkan pesan dengan resolusi gambar yang bervariasi dan jumlah karakter pesan 400 karakter dengan panjang kunci yang bervariasi dan merupakan karakter yang berbeda. Hasil dari pengujian yang ketiga dapat dilihat pada grafik gambar 7 berikut.



Gambar 7. Grafik nilai PSNR pengujian yang ketiga

Dari grafik tersebut dapat dilihat bahwa jumlah panjang kunci dengan karakter yang berbeda berpengaruh sangat sedikit terhadap nilai PSNR yang dihasilkan atau dengan kata lain penurunan kualitas yang terjadi dari gambar yang dihasilkan tidak terlalu berbeda. Hal ini dapat ditandai dengan perubahan nilai PSNR yang sangat sedikit yang dihasilkan oleh masing-masing gambar.

Pengujian keempat yang dilakukan adalah untuk mengembalikan pesan yang sudah disisipkan ke dalam gambar penampung dengan menggunakan sandi yang sama dan sandi yang berbeda ketika dilakukan proses penyisipan. Adapun gambar penampung yang digunakan dalam proses pengujian ini adalah gambar lenna.bmp dan jumlah karakter pesan 400 karakter dengan panjang kunci 5 karakter, yaitu kunci.

Proses selanjutnya adalah mengembalikan pesan yang sudah disisipkan ke dalam gambar penampung dengan menggunakan sandi yang berbeda ketika dilakukan proses penyisipan. Dari hasil proses pengambilan pesan dengan kunci yang berbeda, pesan yang dimunculkan dari proses tersebut tidak sama dengan pesan ketika sebelumnya atau dengan kata lain terjadi perubahan pesan ketika proses pengambilan pesan berlangsung. Meskipun jumlah karakter pesan sama dengan pesan sebelum disisipkan ke dalam gambar, tetapi isi dari pesan tersebut berbeda dari pesan sebelum disisipkan ke dalam gambar.

4. KESIMPULAN

Dari hasil pengujian yang telah dilakukan, maka dapat diambil kesimpulan sebagai berikut :

1. Pemanfaatan algoritma *Blowfish* dan algoritma *Sequential Colour Cycle* untuk penyembunyian pesan pada media gambar berhasil dilakukan.
2. Jumlah karakter pesan yang disisipkan pada setiap gambar penampung berpengaruh terhadap nilai PSNR yang dihasilkan atau dengan kata lain semakin banyak karakter pesan yang disisipkan ke dalam gambar panampung, maka semakin berkurang juga kualitas dari gambar yang dihasilkan.

3. Jumlah panjang kunci dengan karakter yang sama yang digunakan untuk menyisipkan pesan ke dalam gambar penampung tidak berpengaruh terhadap nilai PSNR yang dihasilkan atau dengan kata lain tidak terjadi perubahan kualitas dari gambar yang dihasilkan.
4. Jumlah panjang kunci dengan karakter yang berbeda yang digunakan untuk menyisipkan pesan ke dalam gambar penampung berpengaruh sangat sedikit terhadap nilai PSNR yang dihasilkan atau dengan kata lain perubahan kualitas yang terjadi dari gambar yang dihasilkan tidak terlalu berbeda.
5. Pesan yang sudah disisipkan ke dalam gambar dapat dibaca kembali dengan menggunakan kunci yang sama.
6. Pesan yang sudah disisipkan ke dalam gambar tidak dapat dibaca kembali dengan menggunakan kunci yang berbeda.
7. Berdasarkan perhitungan nilai PSNR tidak didapat nilai dibawah 30 dB yang menunjukkan bahwa kualitas gambar cukup baik dan secara kasat mata tidak terlihat perubahan yang terjadi pada gambar.

5. SARAN

Adapun saran yang dapat diberikan untuk pengembangan sistem lebih lanjut adalah sebagai berikut :

1. Algoritma *Blowfish* dan algoritma *Sequential Colour Cycle* dapat dikombinasikan lagi dengan algoritma lain agar diperoleh hasil yang lebih baik dan aplikasi yang lebih handal, seperti penambahan algoritma kompresi gambar agar gambar yang diperoleh dari aplikasi tersebut memiliki ukuran yang lebih kecil.
2. File yang ingin disisipkan ke dalam media gambar tidak hanya file *.txt saja tetapi dapat berupa file lainnya seperti file audio, video, dan sebagainya.

DAFTAR PUSTAKA

[1] Singh, A., dan Malik, S., 2013, Securing Data by Using Cryptography with Steganography, *International Journal of Advanced Research in Computer Science and Software Enginering*, Vol. 3, 404-409.
[2] Por, L. Y., Beh, D., Ang, T. F., Ong, S. Y., 2013, An Enhanced Mechanism for Image Steganography Using Sequentia Colour Cycle Algorithm, *The International Arab Journal of Information Technology*, Vol. 10, 51-60.
[3] Schneier, B., 1996, *Applied Cryptography*, Ed. 2, Wiley.